Iron Bank Flow Diagram

Iron Bank Steps

1.) Hardening/Dependency Download

1a.) Contributor updates application

1b.) Contributor submits a feature branch to Gitlab. The branch will include the Download.json, Dockerfile, ReadMe, and a license. Once the branch is ready, the Contributor will submit a pull request to the Development Branch.

1c.) Iron Bank Container Hardeners will review the pull request with eyes on code. Once the hardeners validate that the pull request meets criteria specified, they will approve the pull request and merge the feature branch into the development branch.

1d.) The action of merging into the development branch will inform a Jenkins server to start orchestrating the pipeline.

1e.) The first Jenkins Runner will have egress to the Internet. It will look at the Download.json file to identify the components necessary to pull the contributors information into the environment securely. Once in the environment signatures and checksums will be validated to ensure providence.

1f.) Each item downloaded will be sent through a Clam AV scan. If there are threats identified, the download will be quarantined.

1g.) If there are no threats detected, dependencies are pushed into a private Nexus server.

2.) Build Container

2a.) After dependencies have been validated, the Jenkins server will start another Jenkins Runner without any egress to perform the build operations.

2b.) The runner will connect to the Nexus server and pull down the scanned dependencies.

2c.) The runner will build the contributor's container without Internet access.

2d.) After a successful build the container is pushed into the Nexus server

3.) Evaluate Container

3a.) After a successful build, the runner will execute an OpenSCAP, Twistlock, and Anchore scans.

3b.) Results of the scans will be uploaded to the Whitelist Generator

3c.) Contributor will connect to the Whitelist Generator and justify any findings from the scans.

3d.) Iron Bank CVE Approvers will review all the justifications submitted and validate the information as accurate and appropriate to satisfy the finding.

4.) Approve Container

4a.) With the findings of the scans validated, the Authorizing Official (AO) or the AO Designated Representative (DR) will review the entire body of evidence and make the decision to approve the container.

5.) Publish Container

5a.) Once approved, the Whitelist Generator will merge the Development Branch into the Master Branch.

5b.) This will trigger the Jenkin server to start a publish pipeline. Another Jenkins Runner without egress will be started to perform the actions.

5c.) The runner will pull the container in, sign the container, generate a checksum, and pull the body of evidence.

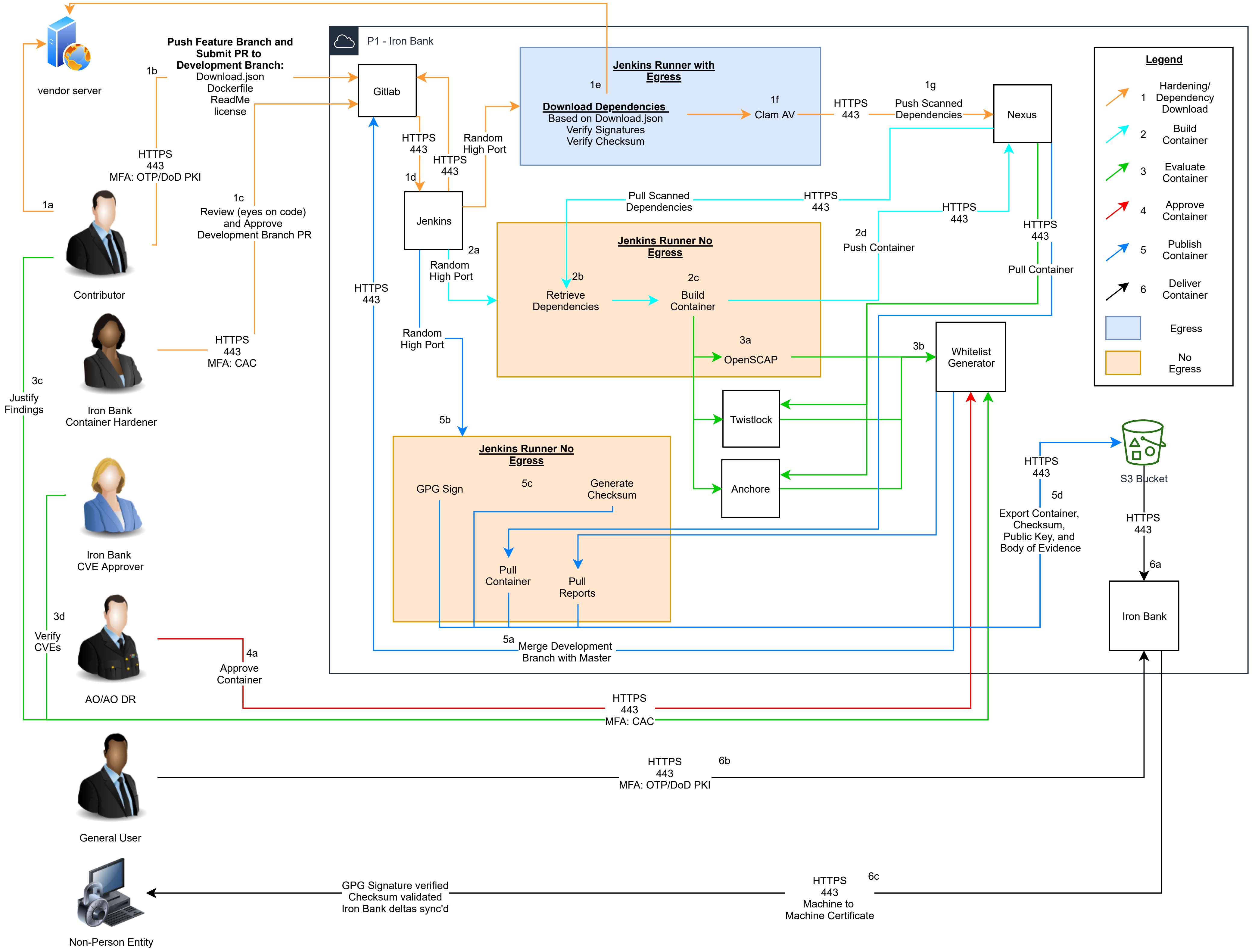
5d.) The package will be pushed to multiple locations segregating the check sum and public key from the container and body of evidence.

6.) Deliver Container

6a.) The Iron Bank web application will retrieve container information from the storage location utilized by the Whitelist Generator.

6b.) Users will be able to access the Iron Bank web application and obtain the body of evidence and containers.

6c.) Platform One environments will have a container that will connect to the Iron Bank authenticating with a machine to machine certificate and synchronize containers.


vendor server
Push Feature Branch and
Submit PR to
Development Branch:
Download.json
Dockerfile
ReadMe
license
1b
HTTPS
443
MFA: OTP/DoD PKI
1a
Contributor
1c
Review (eyes on code)
and Approve
Development Branch PR
HTTPS
443
MFA: CAC
Iron Bank
Container Hardener
3c
Justify
Findings
Iron Bank
CVE Approver
3d
Verify
CVEs
AO/AO DR
4a
Approve
Container
HTTPS
443
MFA: CAC
General User
HTTPS
443
MFA: OTP/DoD PKI
6b
Non-Person Entity
GPG Signature verified
Checksum validated
Iron Bank deltas sync'd
HTTPS
443
Machine to
Machine Certificate
6c
P1 - Iron Bank
Gitlab
Jenkins Runner with
Egress
1e
Download Dependencies
Based on Download.json
Verify Signatures
Verify Checksum
1f
Clam AV
HTTPS
443
1g
Push Scanned
Dependencies
Nexus
HTTPS
443
1d
Random
High Port
HTTPS
443
Jenkins
2a
Random
High Port
Pull Scanned
Dependencies
HTTPS
443
HTTPS
443
2d
Push Container
HTTPS
443
Pull Container
Jenkins Runner No
Egress
2b
Retrieve
Dependencies
2c
Build
Container
3a
OpenSCAP
3b
Whitelist
Generator
Twistlock
Anchore
HTTPS
443
Random
High Port
5b
Jenkins Runner No
Egress
GPG Sign
5c
Generate
Checksum
Pull
Container
Pull
Reports
5a
Merge Development
Branch with Master
HTTPS
443
5d
Export Container,
Checksum,
Public Key, and
Body of Evidence
S3 Bucket
HTTPS
443
6a
Iron Bank
Legend
1
Hardening/
Dependency
Download
2
Build
Container
3
Evaluate
Container
4
Approve
Container
5
Publish
Container
6
Deliver
Container
Egress
No
Egress